unroll

```
int H_out = H - K + 1;
int W_out = W - K + 1;
for (int b = 0; b < B; ++b)
for (int c = 0; c < C; ++c) { int
w_base = c (KK); for (int p = 0; p
< K; ++p)
for (int q = 0; q < K; ++q) {
for(inth=0;h< H_out;++h)
for (int w = 0; w < W_out; ++w) {
int w_unroll = w_base + p * K + q;
int h_unroll = h * W_out + w;
X_unroll[b, h_unroll, w_unroll] =
X[b, c, h + p, w + q];
```

Unroll

Unroll size C*K*K x H_out*W_out

Input (/ for rep) C*(H_out+K-1)*(W_out+K-1)

W forward

```
int m = blockIdx.x;
int h = blockIdx.y / W_grid +
threadIdx.y;
int w = blockIdx.y % W_grid +
threadIdx.x;
float acc = 0.;
for(intc=0; c<C;c++){
for(intp=0;p<K;p++) KxK filter for
(int q = 0; q < K; q++)
            acc += X[c, h + p, w +
q] * W[m, c, p, q];
}
    Y[m, h, w] = acc;
```

Convo matr

```
int X_tile_width = TILE_WIDTH + K-
1;
extern __shared__ float shmem[];
float* X_shared = &shmem[0];
float W_shared =
&shmem[X_tile_width X_tile_width];
m = blockIdx.x;
h_base = (blockIdx.z / W_grid)
TILE_SIZE; the block w_base =
(blockIdx.z % W_grid) TILE_SIZE; x
= threadIdx.x; ty = threadIdx.y; h
= h_base + tx; w = w_base + ty;
float acc = 0.;
for (c = 0; c < C; c++)
tx and ty used as shorthand for
threadIdx.x and threadIdx.y
if (( ty < K) && ( tx < K))
W_shared[ty, tx]= W [m, c, ty, tx];
__syncthreads();
for (int i = h; i < h_base +
X_tile_width; i += TILE_WIDTH) {
for (int j = w; j < w_base +
X_tile_width; j += TILE_WIDTH)
X_shared[i - h_base, j - w_base] =
X[n, c, i,
j]}__syncthreads();}Y[n, m, h, w] =
acc;
```

Calculations

T/B 1024	warps 32 T
Warps 32 T	SM 8 B 1536 T

By **moutaz**
cheatography.com/moutaz/

Published 12th December, 2017.
Last updated 12th December, 2017.